USING THE MD SERIES PLC WITH A THERMISTOR AS A TEMPERATURE SENSOR

INTRODUCTION

This application note will show how to use any MD Series PLC as a device to measure and display temperature using a thermistor temperature sensor. The main topics covered will be: Choosing the senor components, designing and wiring the system, and the program components.

The topics will be discussed in the order stated above because it the logical order of progression to design a temperature sensing system of this type. However, the main focus will be on the last two sections: Design and wiring the system, and the program components.

At the end of this application note there will be a brief description of some of the possible applications of this system as well as a links section for additional information on some topics. Also included with this application note is the sample program that I will reference in this document. The sample program will be fully functional and ready to use for your application. It is free for anyone who has purchased our product and can be used for both industry and hobby.

NOTE: This application note will focus on the Precon Type III Thermistor and using it to measure indoor temperature. Any specifications referenced will apply to this thermistor and the indoor environment and are not necessarily accurate for any thermistor and any environment.

TABLE OF CONTENTS

DESIGNING THE SYSTEM

INTRODUCTION

To design the system, a number of steps must be taken:

1. Choose the sensor components
2. Design the resistor-divider network
3. Get the analog to digital converter (ADC) values
4. Create the lookup table

These steps will be broken down and explained in the next 4 sections.

Before I go into the details of each step, I will explain here the general picture of how the system will work when it is complete. This will make it easier to understand the need for each step. The idea is that the thermistor will sense a temperature that equates to a certain resistance. This resistance will be part of a resistor-divider network that will determine an output voltage based on a constant input voltage. This output voltage is the signal that is input to the PLCs ADC, which will be converted to a 12-bit number inside the PLC. Using a lookup table that contains a range of 12-bit numbers representing temperatures and using some math, this number will be converted into a temperature. This temperature is then displayed on an LCD screen.

CHOOSING SENSOR COMPONENTS

INTRODUCTION

In this section the basics of choosing the right components to measure temperature in a particular environment based on a couple of factors will be discussed. The components that need to be chosen are the:

- Thermistor Sensor
- Thermistor Enclosure

The factors to be considered when choosing a thermistor sensor and enclosure that will be dicussed are:

- Temperature range to measure
- Environmental conditions

The main focus will be on choosing components for a thermistor that measures indoor temperature. Specifically, the Precon Type III thermistor will be discussed.

Please note that there are many possible factors to consider when choosing the right components for an application. Each application will have it's own requirements and this application note will not cover them all, nor will it go into much detail. It will be assumed that you already have knowledge about thermistors and if you don't that you will be learning about them from different sources. The point of this application note is to show how a specific thermistor can be used with our MD Series PLC to measure temperature in an indoor environment.

CHOOSING THE THERMISTOR

When choosing a thermistor sensor and enclosure for any application, there are two main factors that need to be considered:

- Temperature Range
- Environmental Conditions

The temperature range of the Precon Type III sensor is -35°F to 240°F. This range will meet the needs of most applications; however, there are many other sensors from many different manufactures that will offer different ranges.

The environmental condition that will be discussed in this application note is indoor air. In this case the sensor will be placed in a dry area with normal indoor temperatures. This is the ideal scenario for a thermistor because they can typically be damaged by moisture, which is one reason that enclosures are necessary. Thermistors could be placed in any environment as long as they have the proper protection (enclosure).

THERMISTOR SENSOR TYPES

There are many different types of thermistor sensors but they can all be divided into two main categories:

- Negative Temperature Coefficient (NTC)
- Positive Temperature Coefficient (PTC)

The most common type of thermistor, and the one that is used here, is an NTC thermistor. An NTC thermistors resistance will decrease as its temperature goes up. A PTC thermistors resistance will increase as its temperature goes up. Figure 1, below, shows standard temperature vs. resistance graphs of an NTC and PTC thermistor. As can be seen from the graphs, the plot is non-linear with a downward or upward curve.

The Sensor that is used for this application note is the Precon Type III NTC Thermistor. This sensor is listed as being 10 000 Ohms @77°F (25°C). To view a complete scale of resistance vs. temperature for this sensor, see the Thermistor temp-resist chart.pdf. This can also be found in the links section at the end of this document.

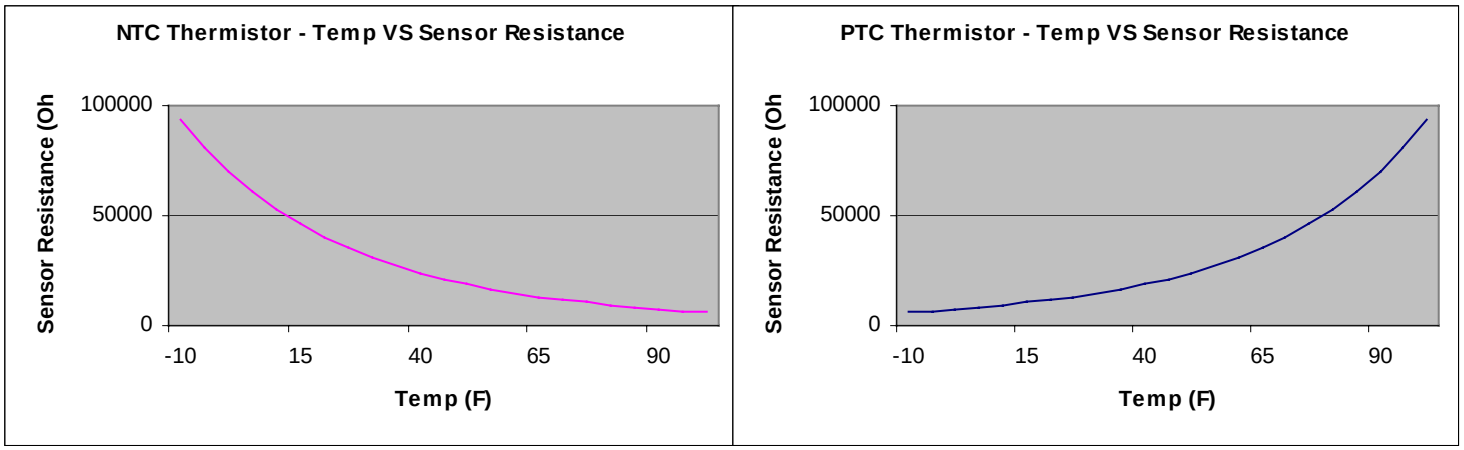

Figure 1: NTC and PTC Thermistor Graphs

THERMISTOR ENCLOSURES

Thermistor enclosures are used to protect the sensors from their environment. There are many different types of enclosures for many different environments. Enclosures range from simple epoxy coating to full plastic and metal casing. Since this application note is focused on sensing indoor air, an enclosure is not necessary. However, in the system being shown, a plastic enclosure that resembles a thermostat is being used. The name of the enclosure-sensor combination is KTR3. This is the data sheet for the KTR3 sensor and enclosure: KTR.pdf. This can also be found in the links section at the end of this document.

RESISTOR-DIVIDER NETWORK

INTRODUCTION

In this section, what a resistor-divider network is and how to choose the resistor will be discussed.

WHAT A RESISTOR-DIVIDER NETWORK IS

A resistor-divider network is a circuit (or portion of a circuit) with two or more resistors and a voltage or current source. The point is to split the voltage in some way so that you have a certain voltage across a resistor. In this case the network consists of an input voltage that is 5V, a thermistor sensor as one resistor, and a standard resistor as the second resistor. This is a common way of interfacing a thermistor sensor to a device that digitally processes the thermistor reading, such as the MD-Series PLC used here. Figure 2, below, is a circuit drawing of a general resistor-divider network.

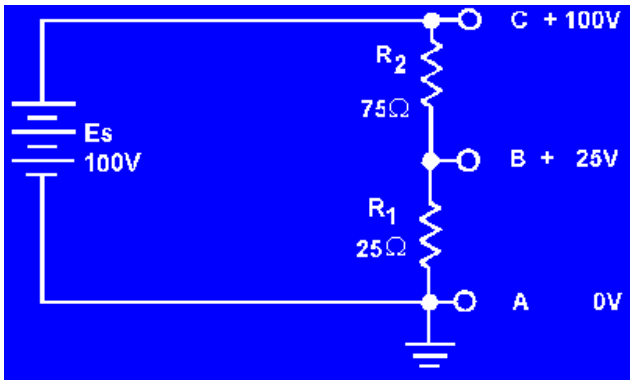

Figure 2: General Resistor-Divider Network

The resistor-divider network that is used in this application works the same way except that the thermistor is not a constant resistance as it changes with temperature. As shown in the circuit below (Figure 3), there is a constant input voltage (5V), a variable resistance (Thermistor), a constant resistance (Resistor), and a variable output voltage (Vadc) that goes to the analog to digital converter (ADC) input on the PLC. As the surrounding temperature changes, the thermistors resistance changes. This changes the resistance ratio, which changes Vadc (the ADC input voltage). The thermistor is already chosen for this application note and the input voltage (Vin) is already chosen to be V, so the only component left to choose is the resistor value. This is discussed in the next section.

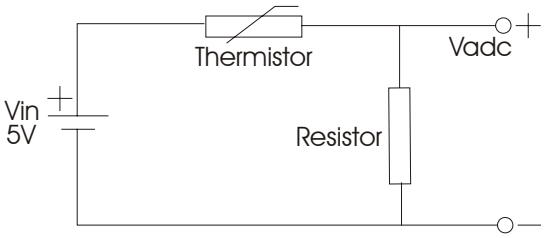

Figure 3: Thermistor Resistor-Divider Network

CHOOSING THE RESISTOR

There are a few different factors to consider when choosing the value of the resistor. The main factors that need to be discussed are: The temperature range to be measured, the voltage range that the ADC will accept, and amount of self-heating that the thermistor can dissipate.

Temperature Range

Each unique temperature has a constant resistance and as the temperature of the sensor goes up, its resistance goes down. A range of temperatures to measure must be pre-determined so that there is a fixed range of resistances to use in future calculations. In this system, the temperature range was decided to be -10°F to 100°F. This covers most outdoor temperatures as well as the indoor air temperatures that this system is designed for. This gives more flexibility for measuring temperatures in different locations and environments.

There are a few different ways to get an exact resistance for a given temperature, including:

1. Complex math
2. Brute force measuring
3. Using a convenient Temperature-Resistance chart provided by the manufacturer.

In this system, choice number 3 is used. This chart provides exact resistances for each whole temperature within the temperature range of the thermistor in increments of 5°F. Click here to view the Temperature-Resistance chart for the Precon Type III thermistor sensor that is used for this system, or find it in the Links section at the end of this document. So we know what the resistance of the sensor is for the temperatures shown between -10°F and 100°F, giving us a minimum sensor resistance and a maximum sensor resistance.

Voltage Range

This is the range of the Vadc voltage that will be the input to the ADC on the PLC. This voltage range is dependant on the previously determined temperature range and the value of the constant resistor in the resistor-divider network. Since the ADC on the PLC accepts voltages in the range of 0-5VDC, the Vadc voltage range must fall within 0-5V.

Now we know the range of resistances, the input voltage, and the range of output voltages, which only leaves the value of the constant resister. From here you would an equation for a resistor-divider

network to calculate a resistor that will keep the Vadc value within that range. Note that Vadc doesn't have to be the full 0-5V range. In this system, Vadc is between 0.6V and 3.5V approximately, which is sufficient to supply an accurate range of ADC values. However, there is more than one resistor value that will give you an accurate range of ADC values. How do you know which one to use, or does it even make a difference? Depending on the resistor value chosen, the resistor-divider network will experience a different amount of power dissipation. This affects the self-heating characteristic of the sensor, which is explained in the next section.

Self-Heating

A thermistor will be able to dissipate a certain amount of power before its temperature is raised 1°F. This is different for every thermistor and it depends on the material/s that its composed of and the manufacturing process. A standard value, and the one used for this system, is 1mW/1°F. This means that for every 1mW of power dissipated in a thermistor, its temperature will be raised by 1°F. Obviously self-heating is bad because it means that the ambient temperature is no longer being measured, only the temperature of the circuit is being measured. Since power needs to be minimized and power is directly proportional to current, it makes sense to try and limit the current as much as possible. The larger the resistance of a circuit the smaller the current and hence, less power is dissipated. In this system the ambient temperature will never change more than 0.5°F due to self-heating.

Final Choice

The resistor used in this system is 15K ohms, which is big enough to limit self-heating while providing a range of voltages that fits the ADCs accepted 0-5V range. This value was basically calculated by trial and error based on the requirements explained previously.

ANALOG TO DIGITAL CONVERTER

The ADC is fairly straightforward. It takes a voltage (Vadc in this case) and converts into a 12-bit number that can be used in the PLCs program. An analog signal of 0V would be 0 when converted and a signal of 5V would be 4092 when converted. Therefore, the full scale range of ADC values is 0 – 4092; However, in this system the full scale is not used, only about 60% of the full scale. This is because of the temperature range limitations and ADC input voltage range limitations. If external circuitry were added, such as a level shifter, then full scale would be possible. If the ADC were full range using the same temperature range, the measurements would be more accurate. However, in this case it is not necessary to be extremely accurate since we are only measuring ambient air temperature. Note that the system is still accurate to the °F when self-heating and range limitations are taken into effect.

LOOKUP TABLE

Each Temperature has a certain sensor resistance associated with it, which determines a certain ADC input voltage, which becomes a certain numeric value. This numeric value being the ADC value between 0 and 4092. Since the graph of temperature vs. resistance is non-linear for the thermistor sensor, there is no simple equation that will accurately calculate the temperature from the ADC value. A common method for calculating data that comes from non-linear measurements is using a lookup table. This is the method used in this system.

A lookup table is a table of data that contains values over a linearly changing unit of measure. The unit of measure could be time, distance, temperature, etc. In this case the unit of measure that data is based on is temperature. This means that for every x°F there is corresponding data such as resistance, ADC voltage, and ADC value. Since the temperature-resistance chart provided by the manufacturer has resistance values for every 5°F, it makes sense to use values in the lookup table that are within the chosen range. In this case the

temperature range is -10°F to 100°F, so the lookup table has data for every 5°F within this range.

For example, the temperature -10°F would have a number of data associated with it. There would be the sensor resistance, the ADC input voltage (Vadc), and the ADC value. It is good to have all of that data when designing the system, but only the ADC value is necessary when building the lookup table for the PLC program. Figure 4, below, shows a picture of the lookup table used to design this system, or click here to see the Excel version of the lookup table.

In this lookup table there is other data shown that is important in designing the system but not important for determining the temperature within the PLC. The data that is important for determining the temperature is highlighted in yellow. There is self-heating data, temperature error data, and constant data such as the values of the static components in the resistor-divider network. All this data is important for design but not for calculating temperature in the PLC.

	Vin	Sensor Res	R	Vout	10-bit ADC	ADC(n) value	Sensor self-heat	Error (F)
	5	10460	15000	2.94579733		(FS=4096)	power (W)	K = 1mW /F
Temp (F)								
-10	5	94070	15000	0.6876318	141	564	0.000198	0.197688
-5	5	81230	15000	0.77938273	160	640	0.000219	0.219298
0	5	70320	15000	0.8790436	180	720	0.000242	0.241500
5	5	61020	15000	0.98658248	202	808	0.000264	0.263971
10	5	53070	15000	1.10180696	226	904	0.000286	0.286337
15	5	46270	15000	1.22409009	251	1004	0.000308	0.308137
20	5	40420	15000	1.35330206	277	1108	0.000329	0.329006
25	5	35390	15000	1.48839055	305	1220	0.000348	0.348443
30	5	31060	15000	1.6283109	333	1332	0.000366	0.366011
35	5	27310	15000	1.77263058	363	1452	0.000381	0.381396
40	5	24060	15000	1.92012289	393	1572	0.000394	0.394250
45	5	21240	15000	2.06953642	424	1696	0.000404	0.404313
50	5	18790	15000	2.2195916	455	1820	0.000411	0.411425
55	5	16650	15000	2.36966825	485	1940	0.000416	0.415534
60	5	14780	15000	2.51846877	516	2064	0.000417	0.416644
65	5	13150	15000	2.6642984	546	2184	0.000415	0.414867
70	5	11720	15000	2.80688623	575	2300	0.000410	0.410388
75	5	10460	15000	2.94579733	603	2412	0.000403	0.403418
80	5	9354	15000	3.07957625	631	2524	0.000394	0.394273
85	5	8378	15000	3.20814441	657	2628	0.000383	0.383235
90	5	7516	15000	3.33096465	682	2728	0.000371	0.370633
95	5	6754	15000	3.44764181	706	2824	0.000357	0.356798
100	5	6078	15000	3.55821235	729	2916	0.000342	0.342012

Figure 4: Lookup Table For Temperatures Between -10°F and 100°F

WIRING THE SYSTEM

INTRODUCTION

In this section, the wiring of the system will be discussed. There are only 5 components involved in the wiring, they are:

1. The power supply
2. The thermistor sensor (with or without enclosure)
3. The resistor-divider network
4. The PLC
5. The LCD display

The wiring of each component will be discussed separately but the complete system wiring is shown in figure 5 below.

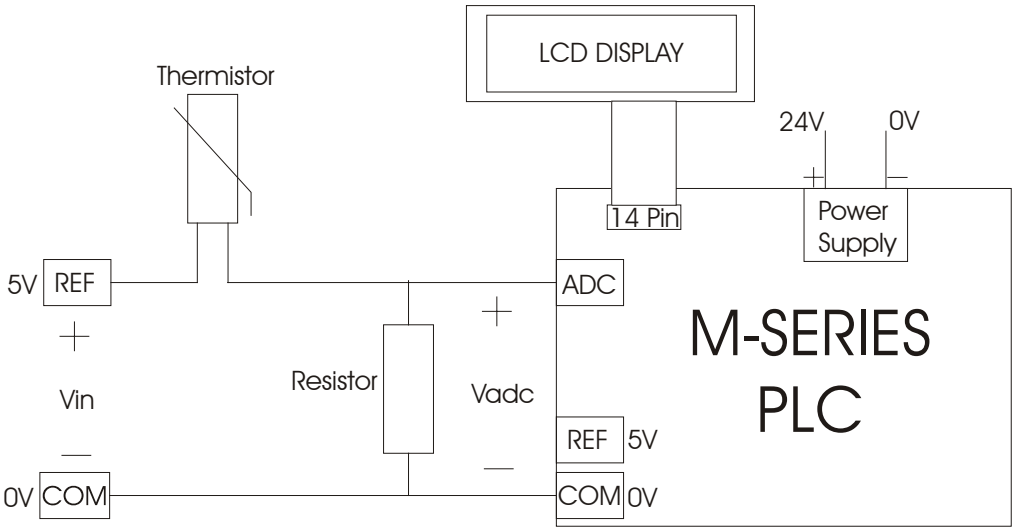

Figure 5: Complete System Wiring

POWER SUPPLY

For information on how to wire the power supply, click here: T100MD888+ Installation Guide.

THERMISTOR WIRING

The thermistor works the same way as a resistor, such that it has two leads and no polarity. Therefore, it can easily be wired into the resistor-divider network as shown above.

RESISTOR DIVIDER NETWORK

The resistor divider network, by itself, involves the thermistor and chosen resistor (15K for this system). The thermistor and resistor are wired in series and polarity is not a factor.

The network is powered from the 5V reference voltage on the PLC. For the T100MD888+ PLC, the reference voltage comes from 3 pins on a DB15 connector as shown below in figure 6. For the T100MD1616+ / T100MD2424+ PLCs, the 5V reference voltage comes from a single input screw terminal on the board that is labeled “+5V”. This 5V reference is actually part of the analog I/O on the PLC and more information on this is in the ADC section below.

ANALOG I/O

The analog I/O are designed slightly different for each of the T100MD Series PLCs. On the T100MD888+, the analog I/O are accessed through a DB15 connector as shown in Figure 6 below. On the T100MD1616+ and T100MD2424+ PLCs, the analog I/O are accessed through a grouping of screw terminals. More information on the analog wiring can be found in the respective installation guides. These are the direct links:

- T100MD888+ Installation Guide
- T100MD1616+ Installation Guide
- T100MD2424+ Installation Guide

Signal	Pin #
A/D #1	8
A/D #2	7
A/D #3	6
A/D #4	5
A/D #5	4
A/D #6	3
A/D #7 or D/A #1	2
A/D #8 or D/A #2	1
Analog Ref. AV_{CC}	13 - 15
Analog ground AV_{SS}	9 - 11

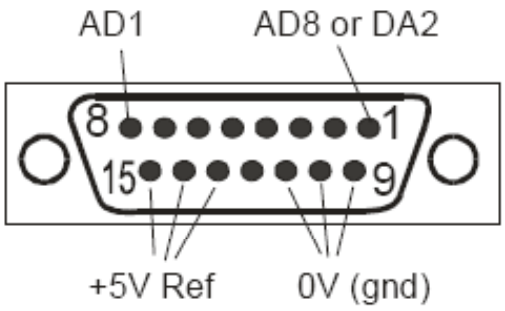

Figure 6: ADC Pinout for T100MD888+ PLC

LCD

The LCD is connected to the PLC through a 14-pin ribbon cable and connector that is just plugged into the LCD on one end and plugged into the PLCs LCD port on the other end. More information on wiring the LCD can be found in the installation guides listed in the "Analog I/O" section above.

PROGRAM COMPONENTS

INTRODUCTION

In this section, the programming portion of the temperature sensor system will be discussed. The program will first be discussed in general, and then it will be broken into sections. The sections are:

- The Lookup Table
- Saving to the EEProm
- Loading From the EEProm
- Interpolating Values

PROGRAM OVERVIEW

When the PLC is first powered up or when it is reset, the program goes through the necessary initialization steps. Next, the PLC reads the temperature from the ADC a designated number of times and takes the average value. Then the measured ADC value is compared against values in the lookup table and the temperature is calculated and displayed on the LCD. After the first program scan, their are no more initializations, so the program cycles through reading the ADC value, calculating the temperature, and displaying the temperature.

LOOKUP TABLE – WHAT IT IS

In this section, what the Lookup Table is will be explained and in a later section, how it is used to determine the correct temperature will be explained.

The Lookup Table is a table that has values associated with addresses from 1001 to 1026. The first 3 values are references that are used for indexing the lookup table and calculating the temperature. The next 23 values (starting from address 1004) are ADC values that represent finite temperatures in increments of 5°F. Figure 7, below, shows the lookup table that is used for the Precon Type III sensor. The values can be put into an excel spreadsheet and that spreadsheet can be converted into a comma separated file (.CSV), which can be put into the PLCs EEProm directly using the EEProm Manager from the I-Trilogi software. This will be explained further in the Initialization section.

1001	-100
1002	50
1003	23
1004	564
1005	640
1006	720
1007	808
1008	904
1009	1004
1010	1108
1011	1220
1012	1332
1013	1452
1014	1572
1015	1696
1016	1820
1017	1940
1018	2064
1019	2184
1020	2300
1021	2412
1022	2524
1023	2628
1024	2728
1025	2824
1026	2916

Figure 7: Lookup Table For Precon Type III Sensor

The address's from 1001 to 1026 are actually data memory (DM[]) locations in the PLC. Each DM[] location holds a value that either corresponds to a reference number (DM[1001] to [1003]) or to an ADC value (DM[1004] to [1026]). As mentioned above, the addresses and their corresponding data can be put directly into EEProm for permanent storage. In this case, the addresses (1001 to 1026) would each be an EEProm address and the corresponding values would be the data held at each EEProm address.

In Figure 8, below, the lookup table is shown again with an extra column that explains what the values are. The first number from the table, -100, is the lowest temperature in the temperature range being used. The value "–100" is actually "-10.0°F" but since the PLC doesn't do floating point arithmetic, the value "-10.0°F" must be represented as "–100". The same is true for the second value, which is 5°F represented as 50. This value of 5°F is the difference in temperature between each ADC value in the lookup table. This goes back to the Temperature-Resistance chart that was provided by the manufacturer of the Precon Type III sensor. The third number is the # of ADC values (Temperatures) in the lookup table. The last 23 numbers are the ADC values that will be used to convert the measured ADC values into temperatures.

Address	Lookup Values	Associated Temperatures and Meanings of Lookup Values
1001	-100	-10 degrees is the lowest temperature and is
		used as a reference for calculating the temperature
1002	50	5 degrees between ADC Values
1003	23	Number of ADC Values
1004	564	-10
1005	640	-5
1006	720	0
1007	808	5
1008	904	10
1009	1004	15
1010	1108	20
1011	1220	25
1012	1332	30
1013	1452	35
1014	1572	40
1015	1696	45
1016	1820	50
1017	1940	55
1018	2064	60
1019	2184	65
1020	2300	70
1021	2412	75
1022	2524	80
1023	2628	85
1024	2728	90
1025	2824	95
1026	2916	100

The ADC values from Address 1004 - 1026 represent these temperatures in degrees F

Figure 8: Lookup Table For Precon Type III Sensor with Explanation of Data

INITIALIZATION

There are two main initializations that need to happen. One is, to save the lookup table values to the EEProm so that they are stored permanently. The other is to transfer the lookup table values from the EEProm to the data memory (DM[]) for quicker and easier access to the table.

SAVING TO THE EEPROM

The first step in the initialization is to save the Lookup Table values to EEProm for permanent storage. This can be done in one of two ways.

1. In the program using a custom function that is activated on the first scan. Every time the PLC is powered on or reset, the values would be saved to EEProm address 1001 to 1026.
2. Using the EEProm manager from the I-Trilogi software to store a comma separated file that contains the EEProm addresses (1001 – 1026) and the corresponding values.

The first option is available but it is only useful for simulation purposes. It does not make sense to use this method for implementation with the PLC because it is a waste of program space and initialization time, and it use EEProm burn cycles every time the PLC is powered on or reset.

The second option is better for implementation with the PLC. The EEProm Manager is a simple tool that comes with the I-Trilogi software and can be accessed from the “Controller” menu in the programming environment. From the “controller” menu, select “EEPROM Manager” and a window will pop-up. This is shown below in Figure 9. This allows you to see what integer and/or string data is already stored in the PLCs EEProm as well as store

integer and/or string data to the PLC. The data that is retrieved from the PLC can be saved to a .CSV file, which can be kept for future use or converted to a spreadsheet in Excel. The data that is to be saved to the PLC can be loaded from a .CSV file, which can be converted from a spreadsheet in Excel. Included with this application note and sample program is a .CSV file form of the lookup table that can be loaded directly into the EEProm using this EEProm manager.

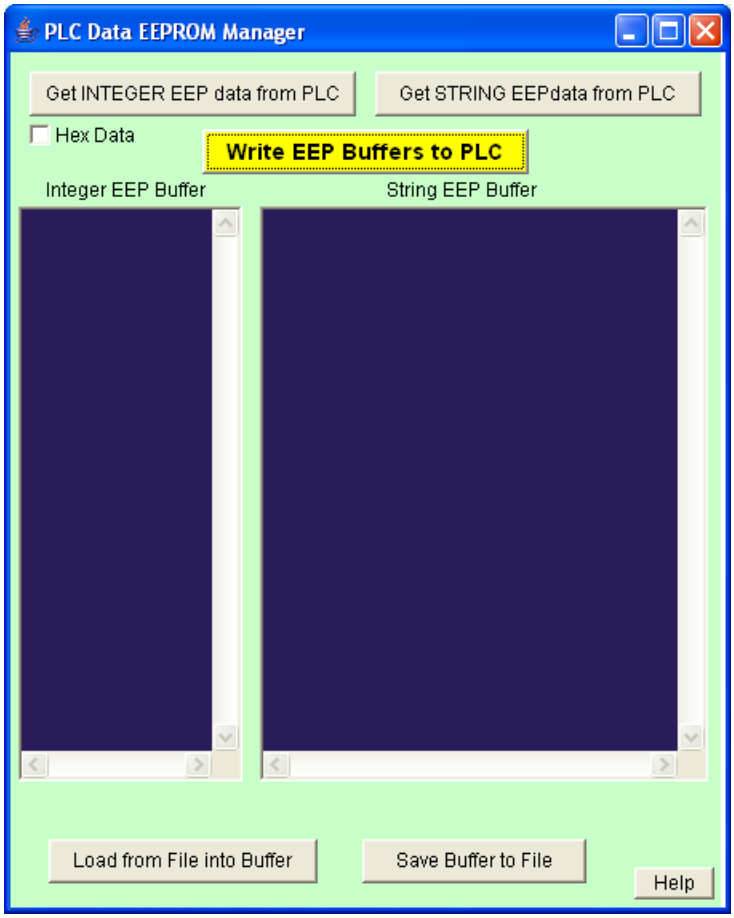

Figure 9: EEPROM Manager

LOADING FROM THE EEPROM

The second part of the initialization process is loading the lookup table from the EEProm into data memory (DM[]). This is done using a custom function that is activated on the first ladder logic scan. There are 4000 DM[] variables that are indexed using the numbers 1 – 4000. The addresses in the lookup table (1001 – 1026) are used to index the DM[] variables and the associated values in the lookup table are the values of DM[1001] – DM[1026]. This makes it easy to cycle through values using loops and to index values numerically. The custom function that loads the table into the DM[] is "init" and its code is shown below.

```
' The thermister data are stored in EEPROM starting from 1001
' address 1001 contains the lowest temperature
' address 1002 contains the increment in temperature reading
' address 1003 contains the number of entries in table (n)
' EEPROM #1004 to 1004+n contains the ADC reading that correspond
' to each temperature
```

```
DM[3903] = LOAD_EEP(1001)                ' lowest temperature
DM[3904] = LOAD_EEP(1002)                ' increment in temperature (degree F)
DM[3905] = LOAD_EEP(1003)                ' number of entries in lookup table

N = DM[3905]                             ' needed by FOR NEXT LOOP

FOR I = 1 to N
  DM[3906+I] = LOAD_EEP(1003+I)          ' load the ADC readings into RAM
NEXT

DM[3906] = (DM[3906+N]-DM[3907])/N       ' average step of ADC reading
```

Code Taken From Custom Function "Init"

READING THE ADC VALUE

Once the initialization process is complete, the next step is to read the current temperature into the ADC as a voltage. This will produce an ADC value that corresponds to a voltage, which corresponds to a resistance, which corresponds to a temperature. Reading the ADC value is quite simple, it involves reading a number of values from the ADC in a short period of time and taking the average of these values in order to obtain a more accurate reading.

The ADC will produce a 12-bit number even though it is only doing a 10-bit conversion. The 10-bit number that is produced by the ADC is converted to a 12-bit number inside the PLCs CPU by multiplying the number by 4. Because of this, the code to interpret the ADC values must handle the values as 12-bit numbers with a range of 0 – 4092 full scale. All of the ADC values in the lookup table (from addresses 1004 – 1026) fall within this range.

The code to take an average of multiple ADC readings is listed below. This is the first part of the ReadTemp custom function.

```
N = 0
FOR I = 1 to 10
  N = N+ADC(1)
NEXT
DM[3901] = N/10              ' get the average of 10 readings to even out noise
```

Code Taken From Custom Function "ReadTemp"

CALCULATING THE TEMPERATURE

After the ADC value is read, the next step is to calculate the temperature from this value. As mentioned in the section "CHOOSING THE THERMISTOR", the graph of Temperature-Resistance for the Precon Type III thermistor (or any thermistor) is non-linear. This means that there is no equation, that the PLC can compute, that will calculate the temperature measured based on the ADC value. This is why the lookup table is used to compare the measured ADC value against known ADC values that equate to temperatures. The code below is taken from the custom function "ReadTemp"; it calls the function "Lookup", which calculates the temperature.

```
CALL Lookup                 ' look up temperature corresponding to value in DM[3901]
                            ' the temperature is returned in DM[3902] by LOOKUP custom function.
                            ' LOOKUP is saved in Function #255
```

USING THE LOOKUP TABLE

In the section "LOOKUP TABLE – WHAT IT IS", the items in the lookup table were discussed. In this section, why a lookup table is used and how it is used will be discussed. The code below shows all of the code in the custom function "Lookup".

```
' ADC reading is passed in DM[3901]
' temperature computed from lookup table will be returned in DM[3902]
' Table starts from DM[3907], number of table entries in DM[3905]
' DM[3905] contains the average ADC increase per entry in LUT
'--------------------------------------------------------------------

IF DM[3901] < DM[3907]                                  ' less than the first entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3907])*DM[3904]/DM[3906]      ' extrapolate from first LUT data
  DM[3902] = DM[3902]+DM[3903]                          ' compute the temperature
  RETURN
ENDIF

IF DM[3901] > DM[3906+DM[3905]]                         ' more than the last entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3906+DM[3905]])*DM[3904]/DM[3906]     ' extrapolate from last LUT data
  DM[3902] = DM[3903]+DM[3904]*(DM[3905]-1)+DM[3902]            ' compute the temperature
  RETURN
ENDIF

I = DM[3901]/DM[3906]-5                                 ' find approximate location to lookup
                                                        ' DM[3906] is the average ADC increment per table entry
IF I < 1: I=1 : ENDIF                                   ' must start from first table entry

IF I > DM[3905]                                         ' temperature is out of range of lookup table
  DM[3902] = DM[3903]+DM[3904]*DM[3905]                 ' return largest temperature of lookup table + 1 increment
  RETURN
ENDIF                                                   ' DATA out of range

WHILE I < DM[3905]                                      ' up to the end of lookup table
  IF DM[3901] < DM[3906+I] GOTO @10
  ENDIF
  I = I+1
ENDWHILE

@10     ' The measured value falls between DM[3906+I-1] and DM[3906+I]

'now compute the interpolated temperature increment from DM[3906+I-1]
DM[3902] = (DM[3901]-DM[3906+I-1])*DM[3904]/(DM[3906+I]-DM[3906+I-1])
DM[3902] = DM[3903]+DM[3904]*(I-2)+ DM[3902]    ' compute actual value
                                                ' DM[3902] contains the return value of this function (degree)
```

Code Taken From Custom Function "Lookup"

Why it is Used

It is used because the relationship between the measured value (resistance / voltage / ADC value) and the calculated temperature is not linear. By creating a table of linearly increasing temperatures with corresponding non-linear ADC values, there is a way to calculate the temperature from the measured ADC value using a combination of comparison and interpolation. This is discussed in the next Section.

How it is Used

The lookup table contains ADC values that have corresponding temperatures that are 5°F different than the previous and next temperature. Since the temperature that is being measured will likely never be the exact temperature in the lookup table, a combination of comparison and interpolation needs to be used. First the measured number will be compared against numbers in the lookup table until it has been placed between to numbers in the table or it is found to out of range of the table. Then, using interpolation, the number of positions that the measured ADC value is away from the base ADC value (corresponding to –10.0°F) is calculated. From this calculation, the temperature can be calculated. This is a general view of the process that is performed inside the "Lookup" custom function. Some code that shows how the "Lookup" custom function determines where the measured ADC value fits into the lookup table or if the ADC value is out of range of the lookup table is shown below.

If the measured ADC value is less than the first entry in the lookup table (1st value = "564" = "–10.0°F"):

```
IF DM[3901] < DM[3907]                    ' less than the first entry of Lookup table (LUT)
```

Code Taken From Custom Function "Lookup"

If the measured ADC value is greater than the last entry in the lookup table (last value = "2916" = "+100°F"):

```
IF DM[3901] > DM[3906+DM[3905]]           ' more than the last entry of Lookup table (LUT)
```

Code Taken From Custom Function "Lookup"

If the measured ADC value is within the lookup table (between "564" and "2916"):

```
I = DM[3901]/DM[3906]-5                   ' find approximate location to lookup
                                          ' DM[3906] is the average ADC increment per table entry
IF I < 1: I=1 : ENDIF                     ' must start from first table entry

WHILE I < DM[3905]                        ' up to the end of lookup table
  IF DM[3901] < DM[3906+I] GOTO @10
  ENDIF
  I = I+1
ENDWHILE

@10     ' The measured value falls between DM[3906+I-1] and DM[3906+I]
```

Code Taken From Custom Function "Lookup"

INTERPOLATING VALUES

Once the measured ADC value has been place between two numbers in the lookup table or has been placed outside of the lookup table, the exact temperature can be calculated in two steps using interpolation. The first step is to calculate the position of the measured ADC value from its nearest value or between its nearest values (depending whether it's in the lookup table or out of range) and then convert that position value into a temperature. The second step is to calculate the temperature that is closest to the temperature corresponding to the measured ADC value.

Depending on whether the measured ADC value is in the lookup table or outside of the lookup table, these two steps are implemented slightly differently but in the same sequence and same general form. These two steps will be explained in more detail in this section in three different scenarios with code examples from the sample program. The first scenario is that the measured ADC value is less than the first value in the lookup table. The second scenario is that the measured ADC value is greater than the last ADC value in the lookup table. The Third scenario is that the measured ADC value is within the lookup table.

Scenario 1 – Measured ADC value is less than the first value in the lookup table

Once the measured ADC value has been placed outside the lookup table and below the first lookup table value, the approximate number of positions that the measured ADC value is from the lowest value is calculated. Then this value is converted into the approximate temperature variation from the base temperature (-10.0°F). This is the first step of interpolation for scenario 1, explained in more detail below.

Each ADC value in the lookup table corresponds to a temperature that is 5°F different than the previous and next temperature. Also, the average difference between ADC values in the lookup table is 102. This means that every time the measured ADC value increases or decreases by an average of 102, the temperature increases or decreases by 5°F. Using this ratio and the difference between the measured ADC value and the first ADC value in the lookup table, the temperature variation from the base temperature (-10.0°F) can be calculated. This is the first step in the two-step interpolation process, which is shown below in the code from the custom function "Lookup". It's the line of code highlighted in green.

```
IF DM[3901] < DM[3907]                          ' less than the first entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3907])*DM[3904]/DM[3906]  ' extrapolate from first LUT data
  DM[3902] = DM[3902]+DM[3903]                  ' compute the temperature
  RETURN
ENDIF
```

Code Taken From Custom Function "Lookup"

The second step is to combine the temperature variation with the base temperature for the exact temperature, as shown in the code below that is highlighted green.

```
IF DM[3901] < DM[3907]                          ' less than the first entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3907])*DM[3904]/DM[3906]  ' extrapolate from first LUT data
  DM[3902] = DM[3902]+DM[3903]                  ' compute the temperature
  RETURN
ENDIF
```

Code Taken From Custom Function "Lookup"

Example:

If the measured ADC value were 100 (< 564, the first LUT value), what would the temperature be?

DM[] LOCATON	DESCRIPTION	VALUE / TYPE
DM[3901]	Measured ADC value	Measured
DM[3902]	Temperature variation and then final temperature.	Calculated
DM[3903]	Base temperature	-100 (= -10.0°F)

DM[3904]	Temperature difference between each LUT value	-50 (= -5.0°F)
DM[3905]	Number of values in lookup table	23
DM[3906]	Average difference between ADC values in LUT	102
DM[3907]	First value in lookup table	564 (= -10.0°F)
DM[3906+DM[3905]]	Last value in lookup table	2916 (= 100°F)

Memory Values Chart

Using the formula to calculate the temperature variation from the base temperature:
DM[3902] = (DM[3901]-DM[3907])*DM[3904]/DM[3906]

DM[3902] = (100 – 564*50/102)
DM[3902] = (100 – 271) ‘where 271 is rounded down because there is no floating point calculations
DM[3902] = (-171)

Now add the temperature variation (-171) to the base temperature (-100) to get the approximated temperature using the formula:
DM[3902] = DM[3902]+DM[3903]

DM[3902] = -171 + (-100)
DM[3902] = -271 ‘the actual temperature is –27.1°F

The temperature would be -27.1°F.

Scenario 2 – Measured ADC value is greater than the last value in the lookup table

Once the measured ADC value has been placed outside the lookup table and above the last lookup table value, the approximate number of positions that the measured ADC value is from the highest value is calculated. Then this value is converted into the approximate temperature variation from the last temperature. This is the first step of interpolation for scenario 2, explained in more detail below.

Each ADC value in the lookup table corresponds to a temperature that is 5°F different than the previous and next temperature. Also, the average difference between ADC values in the lookup table is 102. This means that every time the measured ADC value increases or decreases by an average of 102, the temperature increases or decreases by 5°F. Using this ratio and the difference between the measured ADC value and the last ADC value in the lookup table, the temperature variation from the last temperature (100°F) can be calculated. This is the first step in the two-step interpolation process, which is shown below in the code from the custom function “Lookup”. It’s the line of code highlighted in green.

```
IF DM[3901] > DM[3906+DM[3905]]                    ' more than the last entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3906+DM[3905]])*DM[3904]/DM[3906]    ' extrapolate from last LUT data
  DM[3902] = DM[3903]+DM[3904]*(DM[3905]-1)+DM[3902]           ' compute the temperature
  RETURN
ENDIF
```

Code Taken From Custom Function “Lookup”

The second step is to add the temperature variation to the last temperature for the exact temperature, as shown in the code below that is highlighted green. The last temperature is calculated by adding the base temperature (-100) and the 5°F per step (50) multiplied by the number of steps (23), which equals 100°F.

```
IF DM[3901] > DM[3906+DM[3905]]                          ' more than the last entry of Lookup table (LUT)
  DM[3902] = (DM[3901]-DM[3906+DM[3905]])*DM[3904]/DM[3906]      ' extrapolate from last LUT data
  DM[3902] = DM[3903]+DM[3904]*(DM[3905]-1)+DM[3902]             ' compute the temperature
  RETURN
ENDIF
```

Code Taken From Custom Function "Lookup"

Example:

If the measured ADC value were 3000 (> 2916, the last LUT value), what would the temperature be?

DM[] LOCATON	DESCRIPTION	VALUE / TYPE
DM[3901]	Measured ADC value	Measured
DM[3902]	Temperature variation and then final temperature.	Calculated
DM[3903]	Base temperature	-100 (= -10.0°F)
DM[3904]	Temperature difference between each LUT value	-50 (= -5.0°F)
DM[3905]	Number of values in lookup table	23
DM[3906]	Average difference between ADC values in LUT	102
DM[3907]	First value in lookup table	564 (= -10.0°F)
DM[3906+DM[3905]]	Last value in lookup table	2916 (= 100°F)

Memory Values Chart

Using the formula to calculate the temperature variation from the base temperature:
DM[3902] = (DM[3901]-DM[3906+DM[3905]])*DM[3904]/DM[3906]

DM[3902] = (3000 – 2916*50/102)
DM[3902] = (3000 – 1429) 'where 1429 is rounded down because there is no floating point calculations
DM[3902] = (1571)

Now add the temperature variation (1571) to the last temperature (-100 + 50 * 22 = 1000) to get the approximated temperature using the formula:
DM[3902] = DM[3903]+DM[3904]*(DM[3905]-1)+DM[3902]

DM[3902] = -100 + 5*22 + 1571 'where the green highlighted part is the last LUT value calculated from the first
DM[3902] = +2571 'the actual temperature is +257.1°F

The temperature would be +257.1°F.

Scenario 3 – Measured ADC value is within the lookup table

Once the measured ADC value has been placed between two numbers in the lookup table, the approximate deviance that the measured ADC value is from the value before it is calculated. Then this value is converted into the approximate temperature variation from the LUT value before it. This is the first step of interpolation for scenario 3, explained in more detail below.

Each ADC value in the lookup table corresponds to a temperature that is 5°F different than the previous and next temperature. Also, the difference between ADC values in the lookup table can be calculated. This difference between ADC values is different at different positions in the lookup table. There are 2 parts to calculating the temperature deviance of the

measured ADC value from the closest smaller ADC value in the lookup table.

1. The ratio of degrees F to the difference between ADC values. The degrees F is always 5°F and the difference between ADC values depends on the location in the lookup table as stated above. Code = DM[3904]/(DM[3906+I]-DM[3906+I-1]).
2. The numeric difference between the measured ADC value and the closest smaller ADC value from the lookup table. Code = (DM[3901]-DM[3906+I-1]).

The temperature deviation, from the lookup table value before it, can be calculated by taking the numeric deviation of the measured ADC value from the lookup table value before it and multiplying it by the ratio of degrees F to the difference between ADC values. This will convert the numeric deviance into the temperature deviance. This is performed by multiplying the code from #1. above by the code from #2. above, as shown in the sample code below.

```
@10     ' The measured value falls between DM[3906+I-1] and DM[3906+I]

  ' now compute the interpolated temperature increment from DM[3906+I-1]

DM[3902] = (DM[3901]-DM[3906+I-1])*DM[3904]/(DM[3906+I]-DM[3906+I-1])
DM[3902] = DM[3903]+DM[3904]*(I-2)+ DM[3902] ' compute actual value

' DM[3902] contains the return value of this function (degree)
```

Code Taken From Custom Function "Lookup"

The second step is to add the temperature variation to the closest smaller temperature, calculated from the lookup table, for the exact temperature, as shown in the code below that is highlighted green. This is done in 2 steps:

1. Calculating the temperature, from the lookup table, that is the closest smaller temperature from the measured temperature. Code = DM[3903]+DM[3904]*(I-2).
2. Adding the previously calculated temperature deviance to the temperature from #1. above. Code = DM[3902].

The temperature from the lookup table that the temperature deviance was calculated from is calculated by adding the base temperature (-100) and the 5°F per step (50) multiplied by the number of steps (I - 2). The exact temperature is calculated by adding the values of #1. and #2. from above, as shown in the green highlighted code from the sample code below.

```
@10     ' The measured value falls between DM[3906+I-1] and DM[3906+I]

  ' now compute the interpolated temperature increment from DM[3906+I-1]

DM[3902] = (DM[3901]-DM[3906+I-1])*DM[3904]/(DM[3906+I]-DM[3906+I-1])
DM[3902] = DM[3903]+DM[3904]*(I-2)+ DM[3902] ' compute actual value

' DM[3902] contains the return value of this function (degree)
```

Code Taken From Custom Function "Lookup"

Example:

If the measured ADC value was 1000 (between 904 and 1004 from LUT) and I was 6 (I is the # of positions that 1004 is from the base value, 564, + 1), what would the temperature be?

DM[] LOCATON	DESCRIPTION	VALUE / TYPE
DM[3901]	Measured ADC value	Measured
DM[3902]	Temperature variation and then final temperature.	Calculated
DM[3903]	Base temperature	-100 (= -10.0°F)
DM[3904]	Temperature difference between each LUT value	-50 (= -5.0°F)
DM[3905]	Number of values in lookup table	23
DM[3906]	Average difference between ADC values in LUT	102
DM[3907]	First value in lookup table	564 (= -10.0°F)
DM[3906+DM[3905]]	Last value in lookup table	2916 (= 100°F)

Memory Values Chart

Using the formula to calculate the temperature variation from the base temperature:
DM[3902] = (DM[3901]-DM[3906+I-1])*DM[3904]/(DM[3906+I]-DM[3906+I-1])

DM[3902] = ((1000 – 904)*50/100)
DM[3902] = (96 / 2) ‘where 452 is not rounded down because there is no remainder
DM[3902] = (48)

Now add the temperature variation (548) to the closest smaller temperature, calculated from the lookup table, (-100 + 50 * (I – 2) = 100) to get the exact temperature using the formula:
DM[3902] = DM[3903]+DM[3904]*(I-2)+ DM[3902]

DM[3902] = -100 + 50*4 + 48 ‘where the green highlighted part is the closest smaller temperature, ‘calculated from the lookup table
DM[3902] = +148 ‘the actual temperature is +46.8°F

The temperature would be +14.8°F.

The three scenarios that were just explained cover the possible situations that the program will have to handle, as far as measuring temperature. Once the temperature has been calculated, the next step is to display it on the LCD. This is explained in the next section.

DISPLAYING TO THE LCD

This part of the program is quite simple, as it only involves one line of code that is shown below.

```
SETLCD 1,1, "Temp ADC1="+STR$(DM[3902]/10)+"."+STR$(ABS(DM[3902] MOD 10))+"    "
```

Code Taken From Custom Function “ReadTemp”

There is 3 parts to the line of code that displays the calculated temperature:

1. Standard text describing the value (temperature in this case). Code = "Temp ADC1="
2. The part of the temperature before the decimal point. Code = STR$(DM[3902]/10)

3. The part of the temperature after the decimal point. Code = STR$(ABS(DM[3902] MOD 10))+" "

These 3 parts are concatenated together using the "+" sign (with no quotes). Parts 2 and 3 come from the calculated number that represents temperature. Since the PLC doesn't do floating-point calculation, the temperature that was calculated has to be split into 2 different numbers. The 2 numbers include:

1. The original number divided by 10, rounded to the next lowest whole number. Code = DM[3902]/10
2. The remainder from the original number divided by 10. Code = ABS(DM[3902] MOD 10)

The code from #1. is self explanatory. The calculated temperature that is stored in DM[3902] is divided by 10 and automatically rounded down to the nearest whole number.
The code from #2. may not be so obvious. The "ABS()" part of the code is an absolute value function. It is necessary because the measured and calculated temperature could be negative and since the remainder is to be displayed as a decimal number, it can't be negative. The "MOD" part of the code is a modular function that calculates the remainder of a number divided by another number. In this case, the number being divided is the calculated temperature (DM[3902]) and the number dividing it is 10.

For example:

If the temperature calculated is 14.8°F, the number, stored in DM[3902], that represents this temperature is 148.
The code STR$(DM[3902]/10) will convert 148 in "14" by dividing by 10 and converting into a string using STR$.
The code STR$(ABS(DM[3902] MOD 10))+" " will convert 148 into "8" by taking the remainder of 148/10 and converting it into a string using STR$.

That concludes the explanation of the sample program and the application note. The next section contains some links for further information on the topics covered in this application note.

```
'This function will take in a number of ADC readings and calculate the average for a more stable reading.
'It will then call the Lookup function that calculates the temperature using the ADC reading in a comparison
'with a lookup table and some interpolation.

N = 0
FOR I = 1 to 10
  N = N+ADC(1)
NEXT
DM[3901] = N/10                ' get the average of 10 readings to even out noise

CALL Lookup                    ' look up temperature corresponding to value in DM[3901]
                                ' the temperature is returned in DM[3902] by LOOKUP custom function.
                                ' LOOKUP is saved in Function #255

SETLCD 1,1, "Temp ADC1="+STR$(DM[3902]/10)+"."+STR$(ABS(DM[3902] MOD 10))+"  "
```

Code Taken From Custom Function "ReadTemp"

LINKS

Thermistor Info
http://www.kele.com/tech/monitor/Temperature/TRefTem4.html

Temperature-Resistance Chart
www.**precon**usa.com/oem/temperature/**Thermistor**%20temp-resist%20**chart**.pdf

Self Heating
www.betatherm.com/selfheating.php
http://www.facstaff.bucknell.edu/mastascu/eLessonsHTML/Sensors/TempSensorsSelfHeat.htm

Thermistor Dissipation Constant
www.betatherm.com/dc.php